Callahan, Fiaschetti, Gubenko, Kanar
ANLY 590 Neural Nets and Deep Learning
Final Project Report

Comparing Image Recognition Methodologies

Introduction

Image recognition has been a key field within computer vision and image processing. Its applications have become increasingly important in our everyday lives, which, in turn, lead to more innovation in the field of image recognition. Our team wanted to take this crucial method and apply it to the Stanford Dog Dataset, described below. To accomplish our goal of building the perfect dog breed classification model, we used both conventional convolutional neural network and transfer learning methods, as well as ensemble models that combined these standalone "handmade" and transfer learning models. We compared each model's accuracy score to determine the best dog breed classification model.

Data Overview

As mentioned earlier, our research relies on the Stanford Dog Dataset, a dataset of dog images prepared by scholars at Stanford University. It was created with the purpose of correcting any limitations with previously available dog datasets. Mainly, this dataset has provided the most diverse and complete set of pictures per dog breed. Overall, the data contain 20,580 images of 120 breeds of dogs with approximately 180 pictures per class. We selected this balanced, robust dataset in order to have the ability to test the effect of altering (i.e shrinking) the training set size. For each breed, the dataset contains images of varying degrees of visibility and diverse positioning of the dogs and other subjects of the picture. This allowed us access to the most diverse and complex possible dataset to compare different image classification methodologies.

Images

As mentioned above, the dataset provided us with images of varying properties which allowed us to train the models to have the most complete understanding of each class. In Figure 1, we present a few images for two classes that highlight how diverse both the dataset and each class's picture representations are.

Figure 1. Select Images for Two Data Classes	
Pug	**Dandie Dinmont**
	

Notably, one of the main challenges with this dataset is that the images are not uniform in their sizes or dimension ratios. This required data preprocessing ahead of training the models. We opted to resize all images to a 200x200(x3, because they are RGB images) dimensionality. The minimal dimension in both directions was 210, and we wanted to minimize downsizing without forcing any dimension to expand. We also wanted to use a number that was conducive to clean application of diverse padding and kernel size options (200 is a multiple of 2, 4, 5, and 8).

Classes

The dataset consists of 120 classes with nearly 180 dog images per breed. This large number of images per class allowed our models to pick up on even slight differences between similar breeds.

Methodologies

In our research, we worked to identify the best model or ensemble of models to use for image classification tasks. We first created two ensembles of "handmade" models (i.e. conventional Keras convolutional neural network models trained from scratch). Because these models performed poorly both individually and jointly, we attempted as an alternative to apply a number of popular transfer learning models and [two] ensembles of these transfer learning-informed models to the dataset in order to identify the best-performing one. Whenever experimenting with the different transfer learning embeddings, we kept all hyperparameters consistent across the four transfer learning models in order to be able to compare relative performance. Each of the transfer learning models was pretrained on the ImageNet dataset with the goal of classifying an input image into 1,000 categories.

Conventional Convolutional Neural Networks

We began our project by training and testing conventional convolutional neural networks on this dataset. We experimented with various model characteristics:

- We trained models with kernel sizes of 2x2, 3x3, 4x4, 5x5, and 8x8
- We trained models with both preserving and non-preserving padding techniques
- We experimented with model depths of everywhere between 3 and 10 layers
- We experimented with hidden layer sizes ranging from 16 to 256 nodes
- We experimented with varying amounts of max pooling
- We experimented with ReLU, linear, and TanH activation functions

In the appendix, we have code that showcases two specific samples from this experimentation. They have stride lengths of 1 in both axes throughout, ReLU activation functions throughout, padding that preserves dimensions throughout, a hidden layer node count sequence of 32-32-64-64-128-128-32, and 2x2 Max Pooling between each set of different-sized hidden layers. The only difference is that one utilized 3x3 kernels throughout, while the other utilized 6x6 kernels throughout. Despite all of these attempts, none of these conventional models could significantly outperform random guessing, with test accuracies all approximately 0.9% after one epoch of training.

We caveat that we witnessed performance that surpassed random guessing when models were trained for multiple epochs. However, because it was not computationally feasible to try to train all of the models created in this project for large numbers of epochs, we standardized at training for one epoch across these conventional models as well as the transfer learning models discussed below, and our results can still provide useful evidence regarding model performance in a high-training-efficiency environment.

Conventional Convolutional Neural Network Ensembles

Although the standalone conventional convolutional networks did not surpass random guessing, we wanted to study the performance of ensembles of these poor performers. Therefore, we trained two ensemble models, each of which consisted of five identical sub-models, each with the following characteristics: hidden layers of 32, 16, 8, and 4 nodes, with kernel sizes of 5x5, 5x5, 3x3, and 3x3, respectively; 2x2 max pooling between each of those layers; padding that preserved dimensionality; Adam optimization methodology; and a 120-dimension output layer with a softmax activation function and categorical cross-entropy loss function (these final parameters were required for our problem). In an attempt to evaluate the efficacy of ensembles, we trained each of these sub-models for 5 epochs, rather than the single epoch used above.

The only difference was that, in one ensemble, each of the sub-models was trained on the full training set, whereas in the other ensemble, each of the sub-models was trained on a different fifth of the training set. Our ensemble models function by obtaining each individual sub-model's 120-dimension softmax prediction for each image, and then performing a logistic regression of that 120x5 softmax-populated space. Figures 2 and 3 display these models' performance.

Figure 2. Ensemble of Sub-Models Trained on Full Training Set

	Loss	Accuracy
Sub-Model 1	4.68	0.031
Sub-Model 2	4.57	0.039
Sub-Model 3	4.53	0.043
Sub-Model 4	4.58	0.036
Sub-Model 5	4.49	0.039
Ensemble		0.074

Figure 3. Ensemble of Sub-Models Trained on Training Set Subsets		
	Loss	**Accuracy**
Sub-Model 1	4.61	0.031
Sub-Model 2	4.60	0.032
Sub-Model 3	4.69	0.035
Sub-Model 4	4.51	0.044
Sub-Model 5	4.55	0.037
Ensemble		0.054

These results demonstrate that ensembling can significantly improve model performance: the ensembles' accuracy exceeded the individual sub-models' accuracies by nearly 4% in the first ensemble and by approximately 2% in the second ensemble. We also observe that the individual sub-models did not suffer from being trained on random subsets of the full training set -- the sub-models in the second ensemble did not flag as compared to those in the first ensemble -- but the ensemble performance did suffer from stunted training sets.

Transfer Learning

After witnessing the relatively poor performance of conventional convolutional neural networks on this dataset described above, we sought to generate improved performance using transfer learning. Below are the four transfer learning embeddings we used and their relative performances, as well as the results of ensembling these individual techniques.

VGG

First, we used a VGG transfer learning model. The VGG model was initially constructed to develop an image processing model. This model was established with VGG weights, a global max 2D pooling layer, and a dense 120-element layer using a softmax activation function. Finally, the model was compiled with an Adam optimizer and a categorical cross-entropy for the loss calculation.

Once compiled, the model had over 14 million total parameters, 61,560 of which were trainable. The results from training and testing the model can be seen below in Figure 4. Overall, we can see from the results that this model did not perform well. It achieved an accuracy of only

0.13 for the test data and 0.03 for the train data, and this despite the fact that the VGG-informed model took more than twice as long to train as the other transfer learning models described below.

Figure 4. VGG Model Results		
	Loss	**Accuracy**
Train	4.8512	0.0330
Test	3.8737	0.1284

ResNet50

Next, we used a ResNet50-informed transfer learning model. The ResNet model developed was quite similar to the aforementioned VGG model. It utilized a global max 2D pooling layer and a dense 120-element layer with a softmax activation function. The model was compiled with Adam optimizer and a categorical cross-entropy for the loss.

Comparing it to the VGG model, this model had over 23 million parameters of which 245,880 were trainable. The results from training and testing the model can be found below in Figure 5. With accuracy scores of 0.269 and 0.505 for train and test data, respectively, the ResNet model definitely outperformed the VGG model in both loss and accuracy. However, we do see that ResNet's cross-entropy loss was much higher than that of the VGG model, suggesting that, although ResNet was correct more often, it also had false confidence in many instances.

Figure 5. ResNet50 Model Results		
	Loss	**Accuracy**
Train	13.2497	0.2688
Test	7.2558	0.5052

EfficientNet

Next, we used the EfficientNet model. It was developed similarly to the aforementioned VGG and ResNet models: it also utilized a global max 2D pooling layer and a dense 120-element layer with a softmax activation function. EfficientNet utilized EfficientNet weights for the pretrained model. Lastly, the model was compiled with Adam optimizer and a categorical cross entropy for the loss.

Notably, EfficientNet's training was performed much more quickly than that of the other transfer learning models. However, the results of this model differed significantly as well, as VGG and ResNet substantially outperformed what we achieved using EfficientNet in terms of accuracy. Similarly to the conventional convolutional neural networks described above, EfficientNet was unable to add significant value to random guessing.

Figure 6. EfficientNet Model Results		
	Loss	**Accuracy**
Train	9.1571	0.0156
Test	9.0473	0.0066

Xception

Finally, we used the Xception transfer learning model. It was developed quite similarly to the aforementioned models. It also utilized a global max 2D pooling layer and a dense 120-element layer with a softmax activation. Xception, of course, utilized Xception weights, as opposed to the VGG, ResNet, and EfficientNet weights. And as we've seen with the other models in our attempt to maintain consistency in evaluating the relative quality of the transfer learning embeddings for this task, the model was compiled with Adam optimizer and a categorical cross-entropy loss function.

Notably, Xception had over 21 million total parameters and 245,880 trainable parameters, similar in number to the ResNet model and much higher than the VGG model's relatively modest 14 million total and 61 thousand trainable parameters. Despite this disparity, both ResNet and Xception models trained much faster than the VGG model. And these higher-parameter, faster-training models outperformed VGG as well, as you can see in Figure 7 below. And Xception substantially outperformed what was seen from any of the other three transfer learning models, even ResNet, in both loss and accuracy.

Figure 7. Xception Model Results		
	Loss	**Accuracy**
Train	2.3434	0.5390
Test	1.2336	0.7204

Transfer Learning Ensemble Models

After creating the four aforementioned standalone models (VGG ResNet50, EfficientNet, and Xception), we wanted to see if we could improve the individual models' performance by utilizing ensemble models. As part of this undertaking, we ran two ensemble models (one informed by VGG, ResNet, and Xception, the other informed by those three in addition to EfficientNet), and you can see results below in Figure 8. Our ensemble models function by obtaining each individual sub-model's 120-dimension softmax prediction for each image, and then performing a logistic regression of that 120x3 or 120x4 softmax-populated space. The general idea of this ensembling effort was that, since the standalone Xception model achieved exceptional accuracy compared to other models, we attempted to improve the overall accuracy by ensembling it with others in two different ensembles.

The first ensemble models consisted of Xception, ResNet and VGG models. Ensembling these three models together successfully improved the models' performance for image classification, producing a test accuracy of 0.771. This resulted in an accuracy score which is 0.05 higher than Xception's, the best standalone model.

The second ensemble model added EfficientNet to the first ensemble model. This model also produced a test accuracy of 0.771. Although the standalone EfficientNet model achieved very low accuracy, we nevertheless hoped that it would add more value as part of an ensemble model. Unfortunately, the EfficientNet model was unable to add any value to the ensemble.

Figure 8. Ensemble Model Results

	Components	Accuracy
Model 1	Xception, ResNet, VGG	0.771
Model 2	Xception, ResNet, VGG, EfficientNet	0.771

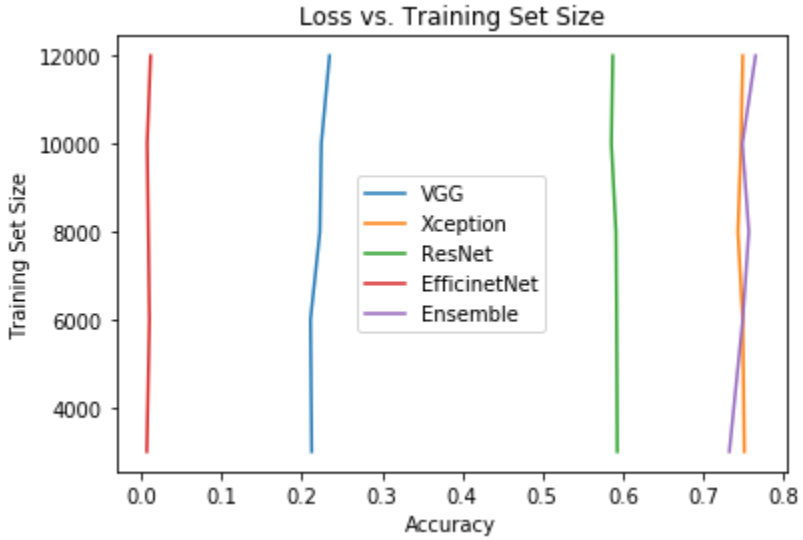
Findings

Comparing the four standalone transfer learning models, the Xception model performed the best in terms of accuracy and loss. As seen in Figure 7, it achieved a test accuracy of 0.7204. This model outperformed the ResNet model (0.5052) and was lightyears ahead of both VGG (0.1284) and EfficientNet (0.0066), despite a training time on par with that of ResNet and EfficientNet and much shorter than that of VGG.

Next, we wanted to see if training set size had any impact on the accuracy each model achieved. We already described how the training set size had no effect on the performance of

conventional convolutional neural networks and only modest effects on the performance of conventional convolutional neural net ensembles, but we wanted to see if this was repeated in the context of transfer learning. The figure below shows the test set loss values for each standalone model trained on dataset sizes of 3,000, 6,000, 8,000, 10,000, and 12,000 (the full training set contained 12,000 images). Notably, the Xception model achieved the smallest loss for any training set size.

The following figure shows the test set accuracy for each standalone and ensemble model (the two ensemble models are identical because the EfficientNet did not add any value).



Interestingly, we can see in the graphs above that the accuracy and loss were not generally altered as the sample size of the training set increased. The accuracy increased modestly and loss decreased modestly in a few cases, but for the majority of the models, it stayed the same. This is

not especially surprising, as moving from approximately 17 observations from each class to approximately 67 observations from each class is not necessarily transformative, but we still expected greater changes in performance. We hypothesize that a significantly larger training set -- in the hundreds of thousands -- likely would have allowed for those increases in performance.

Conclusions and Further Research

Overall, identifying the best model for image classification tasks using the Stanford Dog Breed dataset was a challenging process. The image files were large and unwieldy, leading to high computational costs and frequent kernel failure. Nevertheless, we were able to conduct our research by first applying standalone conventional convolutional and transfer learning models to image classification. We used four transfer learning embeddings - VGG, ResNet, EfficientNet and Xception. As mentioned earlier, the transfer learning techniques outpaced the conventional networks, and the Xception model performed the best, achieving a test accuracy score of 0.720.

Next, we attempted to improve the results by using ensemble models of the aforementioned standalone models. In the case of convolutional neural networks, performance was improved from an accuracy of approximately 3.5% to an accuracy of 7.4%. In the case of transfer learning, we used two different ensemble models, and a model containing Xception, ResNet and VGG improved on Xception's performance by 5%, achieving the best overall accuracy, of 0.771.

Even though we did achieve a well-performing model, there is still room for improvement and further research. More computational capacity certainly allows for improved performance of conventional convolutional nets, and it would likely allow for improvements in the transfer learning models as well. We also assess that the lack of improvement in model performance ushered in by increasing training set size as described in this study was attributable to the entire training set being relatively small. Thus, we would like to explore the possibility that increasing the set of images for each breed to at least 1,000 observations, for a total training set size in the hundreds of thousands, would help us train the model better and achieve better results. And thus we face the need for more and more labeled data, the oft-repeated refrain in the world of neural networks.

Sources & References

Sources

1. Stanford Dog Dataset: http://vision.stanford.edu/aditya86/ImageNetDogs/

References

1. Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao and Li Fei-Fei. Novel dataset for Fine-Grained Image Categorization. First Workshop on Fine-Grained Visual Categorization (FGVC), IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011.
2. Chengwei, N.D., *How to do Transfer learning with Efficientnet, from* https://www.dlology.com/blog/transfer-learning-with-efficientnet/